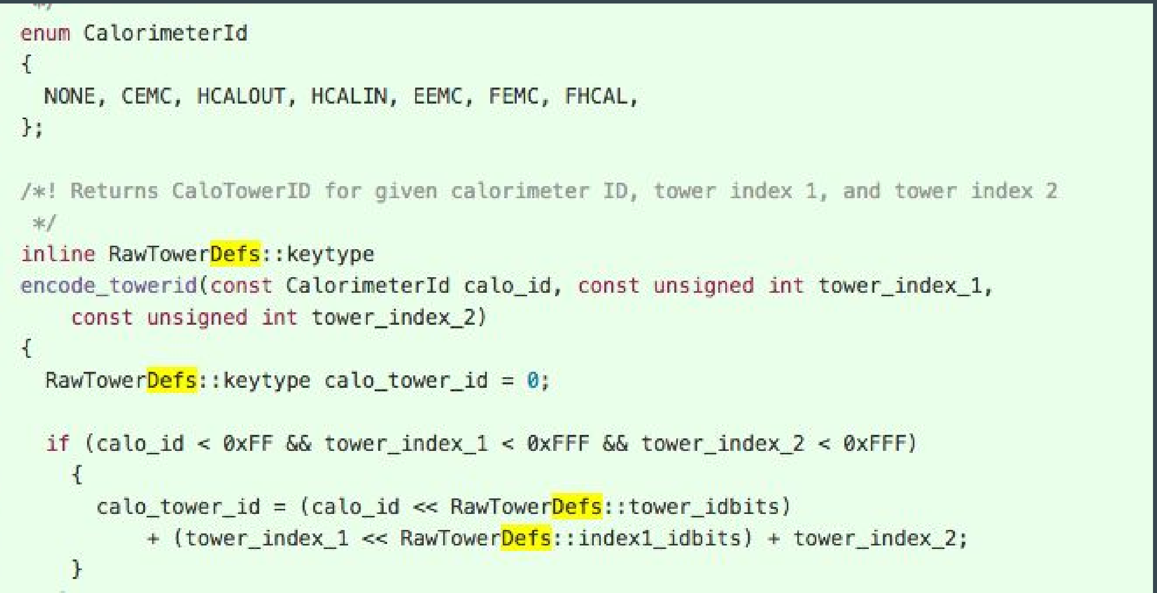
Generic Calorimeter Tower

•••

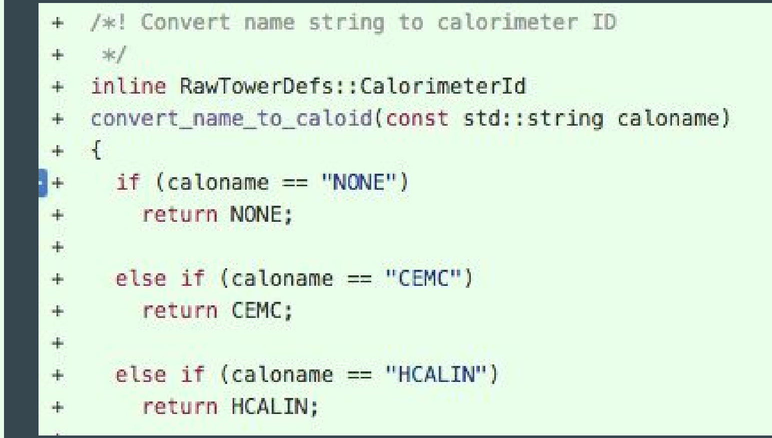
Nils Feege, Jin Huang
November 17, 2015

Objective:
A generalized calorimeter description, which supports both forward and central calorimeters.

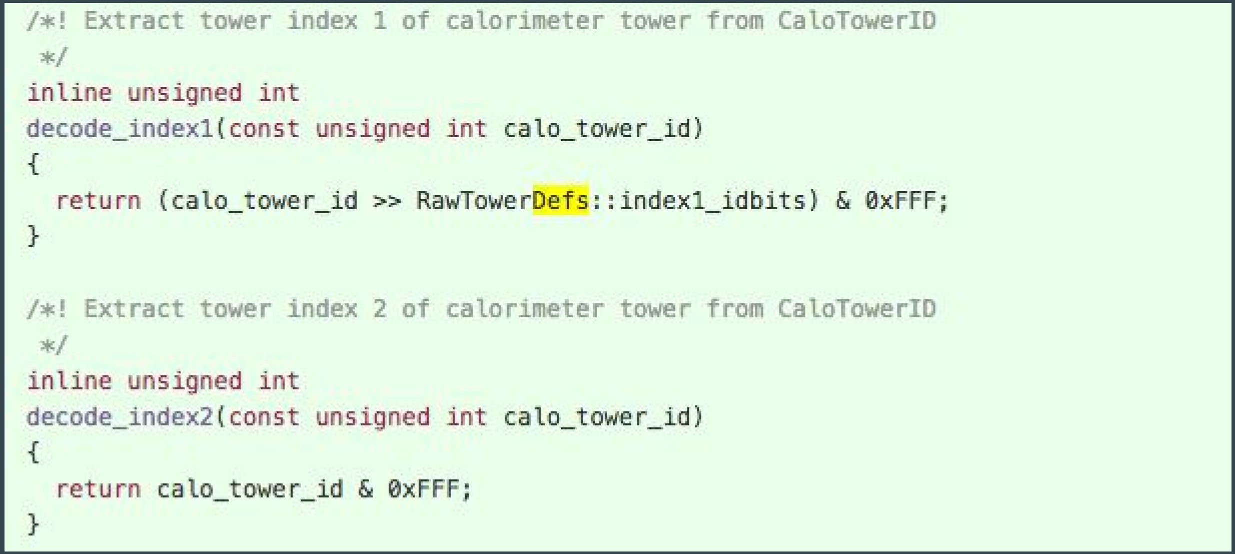
g4cemc/RawTowerDefs.h



```
enum CalorimeterId
{
  NONE, CEMC, HCALOUT, HCALIN, EEMC, FEMC, FHCAL,
};

/*! Returns CaloTowerID for given calorimeter ID, tower index 1, and tower index 2
 */
inline RawTowerDefs::keytype
encode_towerid(const CalorimeterId calo_id, const unsigned int tower_index_1,
    const unsigned int tower_index_2)
{
  RawTowerDefs::keytype calo_tower_id = 0;

  if (calo_id < 0xFF && tower_index_1 < 0xFFF && tower_index_2 < 0xFFF)
    {
      calo_tower_id = (calo_id << RawTowerDefs::tower_idbits)
          + (tower_index_1 << RawTowerDefs::index1_idbits) + tower_index_2;
    }
```

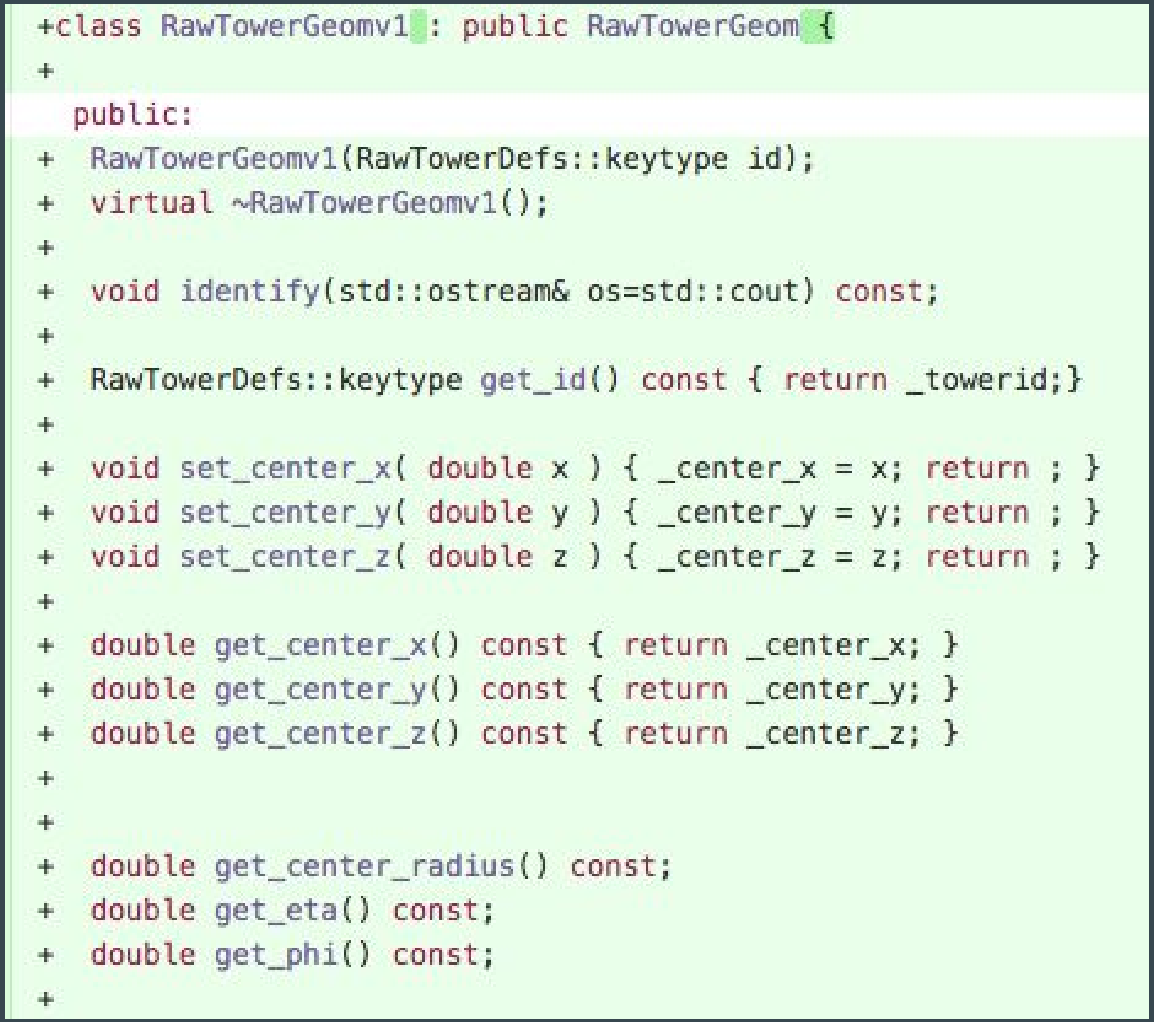


```
+  /*! Convert name string to calorimeter ID
+   */
+  inline RawTowerDefs::CalorimeterId
+  convert_name_to_caloid(const std::string caloname)
+  {
+    if (caloname == "NONE")
+      return NONE;
+
+    else if (caloname == "CEMC")
+      return CEMC;
+
+    else if (caloname == "HCALIN")
+      return HCALIN;
```



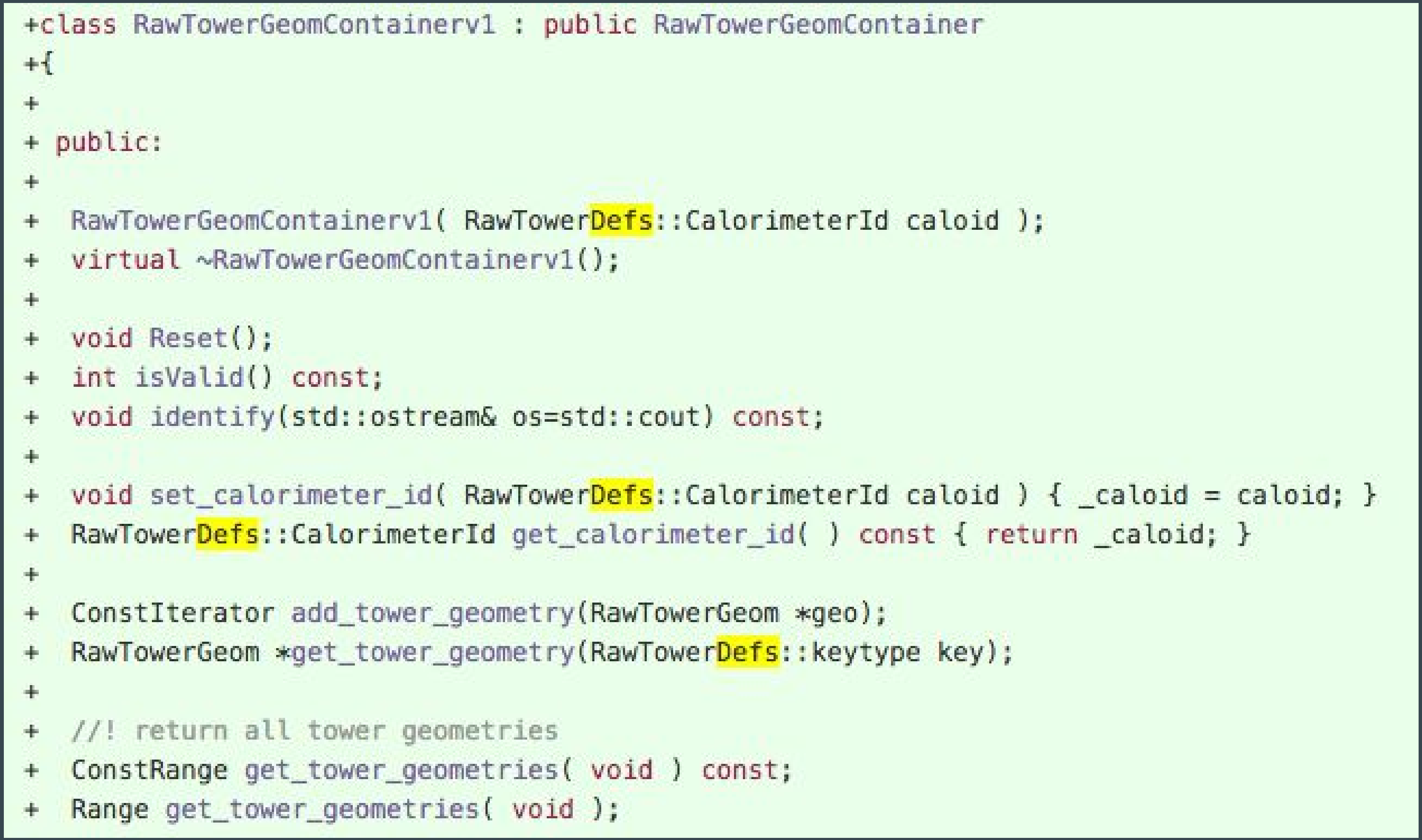
```
/*! Extract tower index 1 of calorimeter tower from CaloTowerID
 */
inline unsigned int
decode_index1(const unsigned int calo_tower_id)
{
  return (calo_tower_id >> RawTowerDefs::index1_idbits) & 0xFFF;
}

/*! Extract tower index 2 of calorimeter tower from CaloTowerID
 */
inline unsigned int
decode_index2(const unsigned int calo_tower_id)
{
  return calo_tower_id & 0xFFF;
}
```

g4cemc/RawTowerGeom.h
g4cemc/RawTowerGeomv1.h



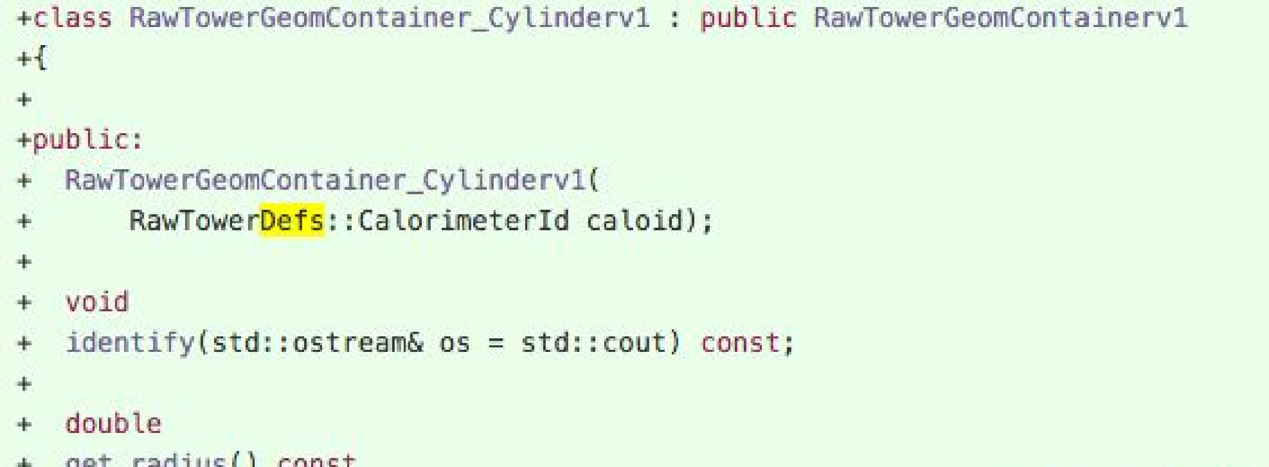
```
+class RawTowerGeomv1 : public RawTowerGeom {
+
  public:
+  RawTowerGeomv1(RawTowerDefs::keytype id);
+  virtual ~RawTowerGeomv1();
+
+  void identify(std::ostream& os=std::cout) const;
+
+  RawTowerDefs::keytype get_id() const { return _towerid;}
+
+  void set_center_x( double x ) { _center_x = x; return ; }
+  void set_center_y( double y ) { _center_y = y; return ; }
+  void set_center_z( double z ) { _center_z = z; return ; }
+
+  double get_center_x() const { return _center_x; }
+  double get_center_y() const { return _center_y; }
+  double get_center_z() const { return _center_z; }
+
+
+  double get_center_radius() const;
+  double get_eta() const;
+  double get_phi() const;
+
```

g4cemc/RawTowerGeomContainer.h
g4cemc/RawTowerGeomContainerv1.h

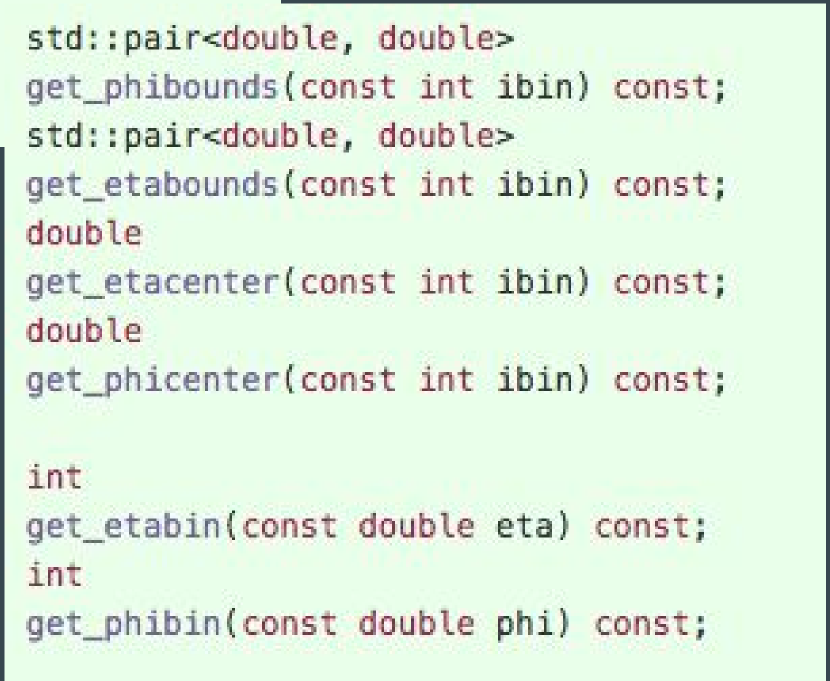


```
+class RawTowerGeomContainerv1 : public RawTowerGeomContainer
+{
+
+ public:
+
+  RawTowerGeomContainerv1( RawTowerDefs::CalorimeterId caloid );
+  virtual ~RawTowerGeomContainerv1();
+
+  void Reset();
+  int isValid() const;
+  void identify(std::ostream& os=std::cout) const;
+
+  void set_calorimeter_id( RawTowerDefs::CalorimeterId caloid ) { _caloid = caloid; }
+  RawTowerDefs::CalorimeterId get_calorimeter_id( ) const { return _caloid; }
+
+  ConstIterator add_tower_geometry(RawTowerGeom *geo);
+  RawTowerGeom *get_tower_geometry(RawTowerDefs::keytype key);
+
+  //! return all tower geometries
+  ConstRange get_tower_geometries( void ) const;
+  Range get_tower_geometries( void );
```

g4cemc/RawTowerGeomContainer_Cylinderv1.h



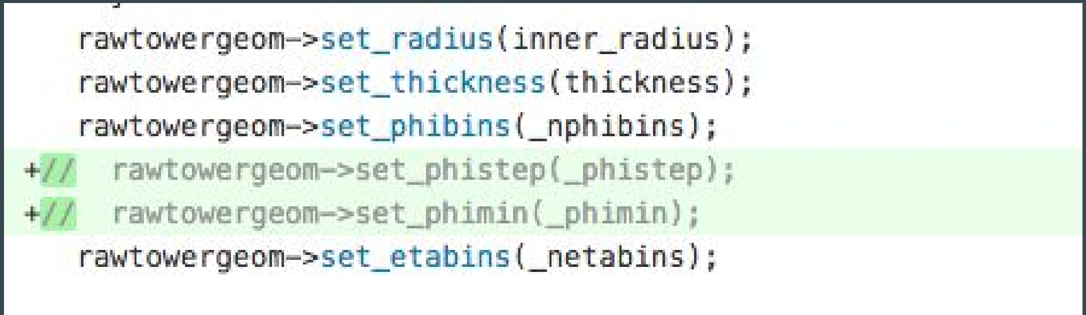
```
+class RawTowerGeomContainer_Cylinderv1 : public RawTowerGeomContainerv1
+{
+
+public:
+  RawTowerGeomContainer_Cylinderv1(
+      RawTowerDefs::CalorimeterId caloid);
+
+  void
+  identify(std::ostream& os = std::cout) const;
+
+  double
+  get_radius() const
+  {
+    return radius;
+  }
```



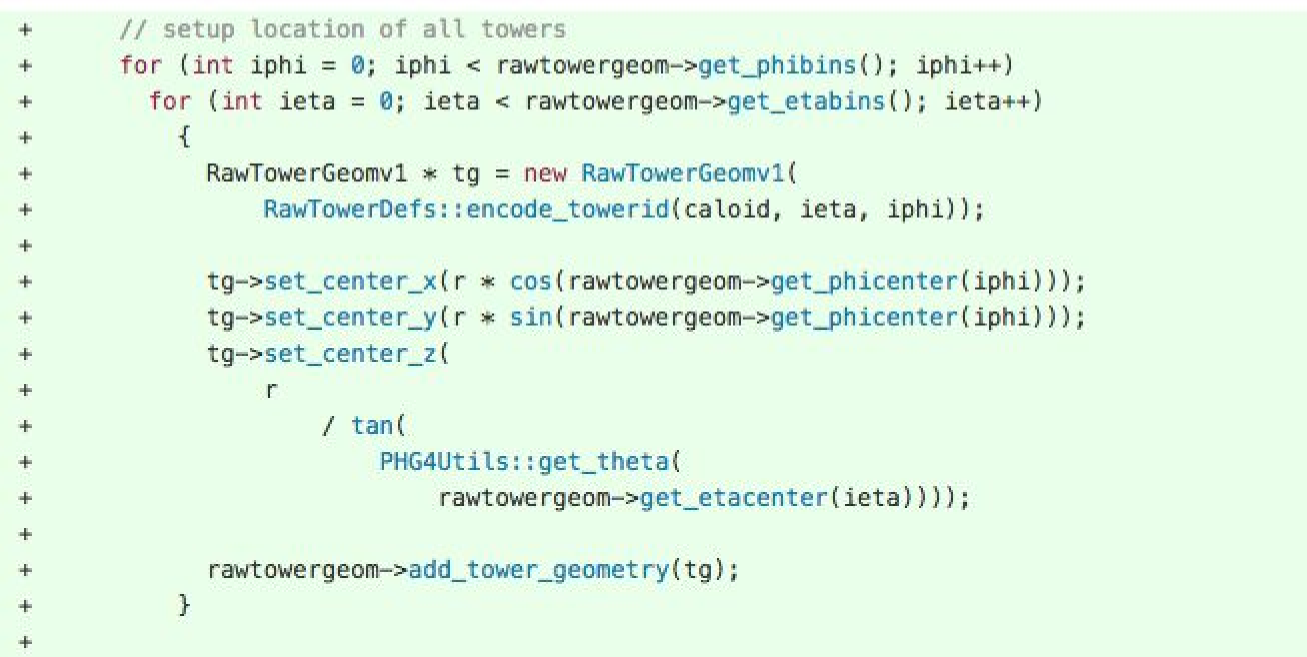
```
std::pair<double, double>
get_phibounds(const int ibin) const;
std::pair<double, double>
get_etabounds(const int ibin) const;
double
get_etacenter(const int ibin) const;
double
get_phicenter(const int ibin) const;

int
get_etabin(const double eta) const;
int
get_phibin(const double phi) const;
```

g4cemc/RawTowerBuilder.cc (Cylinder calorimeter)

The tower builder classes also construct the tower geometries.



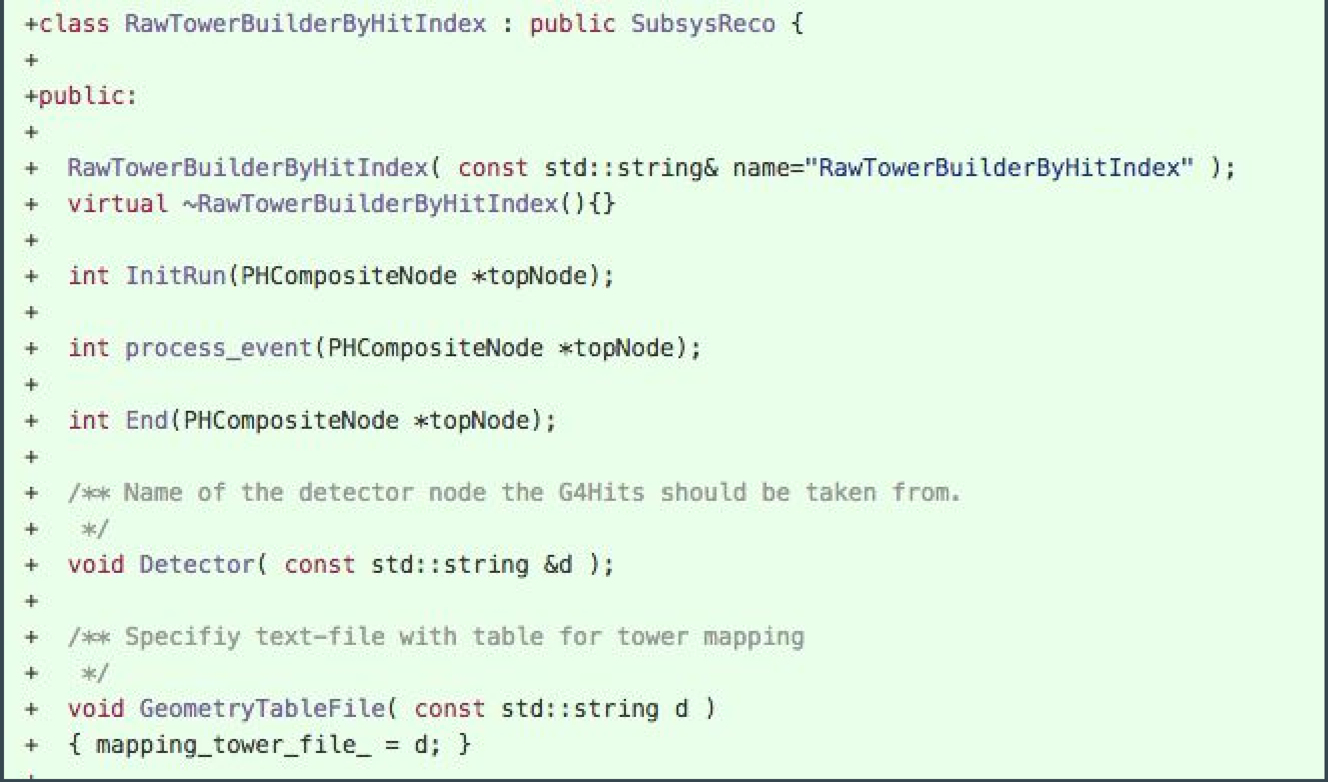
```
   rawtowergeom->set_radius(inner_radius);
   rawtowergeom->set_thickness(thickness);
   rawtowergeom->set_phibins(_nphibins);
+//  rawtowergeom->set_phistep(_phistep);
+//  rawtowergeom->set_phimin(_phimin);
   rawtowergeom->set_etabins(_netabins);
```



```
+      // setup location of all towers
+      for (int iphi = 0; iphi < rawtowergeom->get_phibins(); iphi++)
+        for (int ieta = 0; ieta < rawtowergeom->get_etabins(); ieta++)
+          {
+            RawTowerGeomv1 * tg = new RawTowerGeomv1(
+                RawTowerDefs::encode_towerid(caloid, ieta, iphi));
+
+            tg->set_center_x(r * cos(rawtowergeom->get_phicenter(iphi)));
+            tg->set_center_y(r * sin(rawtowergeom->get_phicenter(iphi)));
+            tg->set_center_z(
+                r
+                    / tan(
+                        PHG4Utils::get_theta(
+                            rawtowergeom->get_etacenter(ieta))));
+
+            rawtowergeom->add_tower_geometry(tg);
+          }
+
```

g4cemc/RawTowerBuilderByHitIndex.h

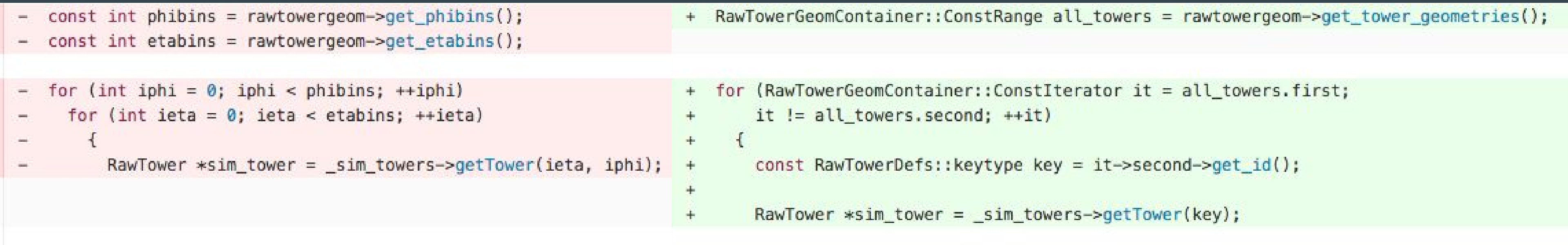
Currently: Read geometry from text file. Possible updates: Let G4Detector module write geometry file? Use database?



```
+class RawTowerBuilderByHitIndex : public SubsysReco {
+
+public:
+
+  RawTowerBuilderByHitIndex( const std::string& name="RawTowerBuilderByHitIndex" );
+  virtual ~RawTowerBuilderByHitIndex(){}
+
+  int InitRun(PHCompositeNode *topNode);
+
+  int process_event(PHCompositeNode *topNode);
+
+  int End(PHCompositeNode *topNode);
+
+  /** Name of the detector node the G4Hits should be taken from.
+   */
+  void Detector( const std::string &d );
+
+  /** Specifiy text-file with table for tower mapping
+   */
+  void GeometryTableFile( const std::string d )
+  { mapping_tower_file_ = d; }
```

g4cemc/RawTowerDigitizer.cc

Updated to work for both cylinder and forward geometries.



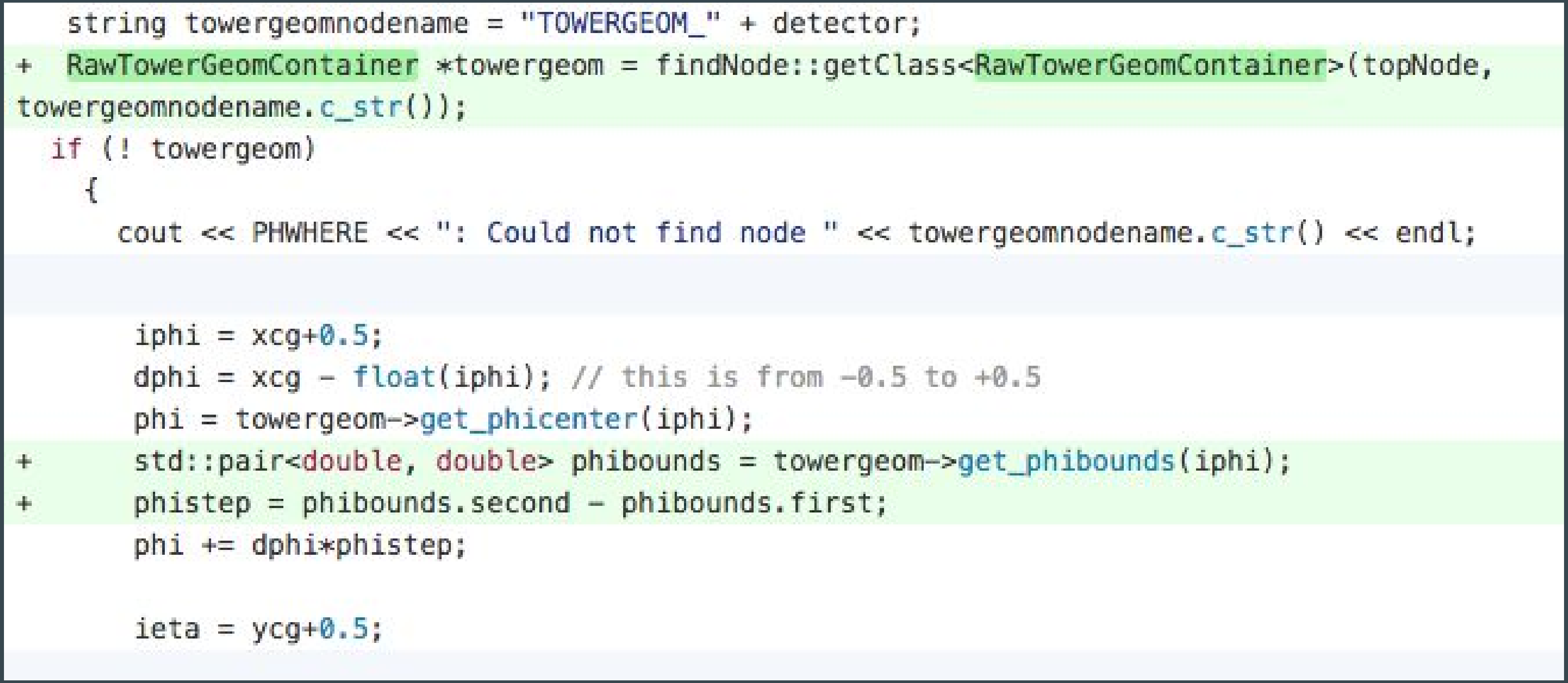
```
-  const int phibins = rawtowergeom->get_phibins();
-  const int etabins = rawtowergeom->get_etabins();

-  for (int iphi = 0; iphi < phibins; ++iphi)
-    for (int ieta = 0; ieta < etabins; ++ieta)
-      {
-        RawTower *sim_tower = _sim_towers->getTower(ieta, iphi);
```

```
+  RawTowerGeomContainer::ConstRange all_towers = rawtowergeom->get_tower_geometries();

+  for (RawTowerGeomContainer::ConstIterator it = all_towers.first;
+      it != all_towers.second; ++it)
+    {
+      const RawTowerDefs::keytype key = it->second->get_id();
+
+      RawTower *sim_tower = _sim_towers->getTower(key);
```

g4cemc/RawClusterBuilderv1.cc

RawClusterBuilder (current version) only works for cylinder geometries-

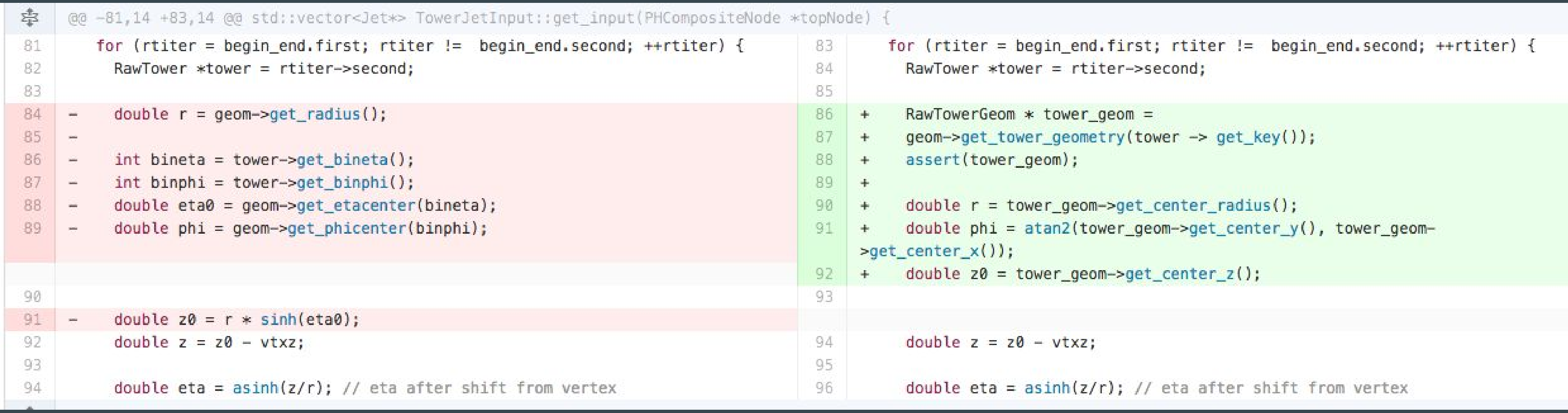
Update? Write separate cluster builder for forward detectors?



```
   string towergeomnodename = "TOWERGEOM_" + detector;
+  RawTowerGeomContainer *towergeom = findNode::getClass<RawTowerGeomContainer>(topNode,
towergeomnodename.c_str());
  if (! towergeom)
    {
      cout << PHWHERE << ": Could not find node " << towergeomnodename.c_str() << endl;


       iphi = xcg+0.5;
       dphi = xcg - float(iphi); // this is from -0.5 to +0.5
       phi = towergeom->get_phicenter(iphi);
+      std::pair<double, double> phibounds = towergeom->get_phibounds(iphi);
+      phistep = phibounds.second - phibounds.first;
       phi += dphi*phistep;

       ieta = ycg+0.5;
```

g4jets/TowerJetInput.C

Now it uses tower x/y/z geometry to calculate eta/phi with the vertex correction, and should be ready to run on forward calorimeters alread



```
@@ -81,14 +83,14 @@ std::vector<Jet*> TowerJetInput::get_input(PHCompositeNode *topNode) {
```

```
    for (rtiter = begin_end.first; rtiter !=  begin_end.second; ++rtiter) {
      RawTower *tower = rtiter->second;

-     double r = geom->get_radius();
-
-     int bineta = tower->get_bineta();
-     int binphi = tower->get_binphi();
-     double eta0 = geom->get_etacenter(bineta);
-     double phi = geom->get_phicenter(binphi);

-     double z0 = r * sinh(eta0);
      double z = z0 - vtxz;

      double eta = asinh(z/r); // eta after shift from vertex
```

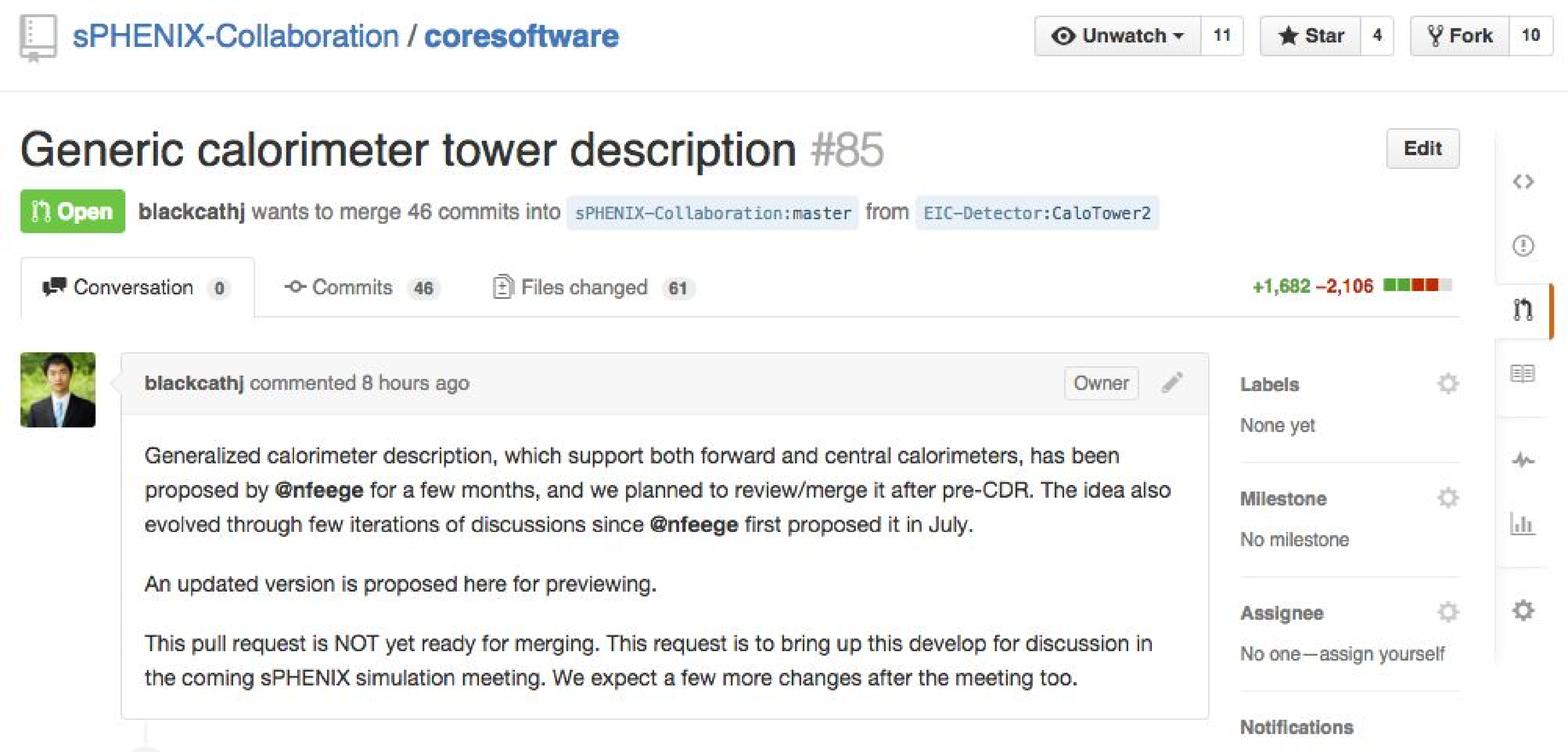
```
    for (rtiter = begin_end.first; rtiter !=  begin_end.second; ++rtiter) {
      RawTower *tower = rtiter->second;

+     RawTowerGeom * tower_geom =
+     geom->get_tower_geometry(tower -> get_key());
+     assert(tower_geom);
+
+     double r = tower_geom->get_center_radius();
+     double phi = atan2(tower_geom->get_center_y(), tower_geom->get_center_x());
+     double z0 = tower_geom->get_center_z();

      double z = z0 - vtxz;

      double eta = asinh(z/r); // eta after shift from vertex
```

Retired Classes in g4cemc/

- CrystalCalorimeterDigitization
- CrystalCalorimeterTowerBuilder
- RawTowerBuilderCone
- RawTowerCombiner
- RawTowerv2

Pull request on GitHub
(NOT ready for merge, REVIEW only)

https://github.com/sPHENIX-Collaboration/coresoftware/pull/85



Generic calorimeter tower description #85

blackcathj wants to merge 46 commits into `sPHENIX-Collaboration:master` from `EIC-Detector:CaloTower2`

blackcathj commented 8 hours ago

Generalized calorimeter description, which support both forward and central calorimeters, has been proposed by **@nfeege** for a few months, and we planned to review/merge it after pre-CDR. The idea also evolved through few iterations of discussions since **@nfeege** first proposed it in July.

An updated version is proposed here for previewing.

This pull request is NOT yet ready for merging. This request is to bring up this develop for discussion in the coming sPHENIX simulation meeting. We expect a few more changes after the meeting too.